Universidad Autónoma de Madrid
Escuela politécnica superior

Grado en Ingeniería Informática

Trabajo Fin de Grado

DESARROLLO DE UN INDEXADOR DE CONTENIDO PARA LA DEEP WEB

Autor: Marina Castellanos Nicolás
Tutor: David Arroyo Guardeño

Junio 2018

DESARROLLO DE UN INDEXADOR DE CONTENIDO PARA LA DEEP WEB

Autor: Marina Castellanos Nicolás
Tutor: David Arroyo Guardeño

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2018

Resumen

Resumen

La cantidad de información contenida en Internet es de gran relevancia cuando nos adentramos en el ámbito de la ciberinteligencia, desde la cual se persigue la creación de inteligencia a través del análisis, evaluación e interpretación de datos recabados de la web. A día de hoy la extracción de información de páginas webs en Internet es una práctica común y refinada, con la que se pretende indexar información con diferentes fines personales y profesionales: ya sean de negocio, estadísticos, de prevención y seguridad, etc. No obstante, hasta hace poco tiempo había una parte de la red a la que no llegaban los rastreadores de Internet: páginas webs que no estaban hiperenlazadas. Para poder llegar a estas páginas webs, que son, entre otras, las que componen la *Deep Web*, se deberá interactuar con formularios webs, por lo que el flujo de funcionamiento se aleja de el de los indexadores que conocíamos.

En primer lugar, este trabajo pretende estudiar desde la base cómo se construye un indexador de contenido. Se va explicar las diferentes partes que lo componen y su funcionamiento. Se identifican los mayores problemas que se pueden encontrar al desarrollar un indexador y se estudian las principales soluciones propuestas en la literatura relacionada.

Seguidamente se pasará al ámbito de la *Deep Web*, en el que se verán las diferencias con respecto a la indexación de contenido de la *Surface Web*. Este trabajo pretende hacer un análisis de las metodologías y herramientas disponibles actualmente con las que se pueda automatizar un rastreador. Por último, se va a llevar a cabo una labor de clasificación de trabajos relacionados, estudiando las metodologías que usan, la estructura de sus rastreadores y soluciones propuestas. El trabajo se presenta como un trabajo de investigación teórico y se planteará para el futuro el desarrollo propio de un indexador de contenido para la *Deep Web*.

Palabras Clave

Deep Web, Dark Net, Tor, indexador, crawler, rellenado de formularios, query template, ciberinteligencia

Abstract

The amount of information in the Internet is of major relevance when we dive into the scope of cyberintelligence, in which the main goal is to create intelligence through the analysis, evaluation and interpretation of data captured from the web. Nowadays, data retrieval from web pages in Internet is a common practice and is a refined practice that many take part in for both personal and professional reasons: these aims could be business oriented, statistical oriented, prevention and defense oriented, etc. However, it hasn't been until recently that web crawlers were missing a certain group of web pages that aren't hyperlinked among each other. In order to reach these sites, which are the ones that constitute the Deep Web, crawlers must learn to interact with web forms, which makes the process of indexing such content different from the one that was known and practiced.

The first goal of this paper is to study throughout how a web crawler and indexer of content is created. The main parts that compose it and how they work will be explained. The main problems that are possible to encounter whilst developing an index system will be presented along with the main solutions proposed by related literature.

Secondly, this paper will focus on the Deep Web and on the differences with Surface Web indexing and crawling. An analysis of the methodologies and tools available to automate a crawling program will be explained. Lastly, there will be a classification of related works done with the taxonomies presented throughout the project. This paper is constituted as a theoretical investigation project and a practical implementation of an indexer of content for Deep Web will be proposed for the future.

Key words

Deep Web, Dark Net, Tor, index, crawler, form filling, query template, cyberintelligence

Agradecimientos

A los que confiaron en mí.

Índice general

Índice de Figuras

1 Introducción

1.1. Motivación del proyecto

Cada minuto The Weather Channel recibe 18 millones de peticiones sobre pronósticos del tiempo. En Snapchat se publican más de 500.000 fotos, y mundialmente se envian más de 103 millones de emails de Spam cada 60 segundos [7]. Toda esta información se crea y se queda almacenada en la red, y la curva de crecimiento no hace más que incrementar.

Con la llegada de la era de la digitalización y la incorporación de aparatos tecnológicos en numerosos ámbitos de nuestro día a día ha llegado también un creciente interés sobre cómo poder acceder a la máxima posible y crear inteligencia partiendo de todo esta cantidad de información. La cultura de la información crece y se asientan en la sociedad las ideas de que no solo el conocimiento es poder[1] , también es un derecho[2]. La digitalización de nuestras vidas y de nuestra sociedad ha cambiado nuestra perspectiva hacia Internet, donde antes eramos cautos y dudábamos de credibilidad de la información ahora vemos fuentes de conocimiento al alcance de nuestras manos [8]. La tendencia demuestra que no solo se plasma este creciente interés por la información en el ámbito personal; cada vez más profesionales buscan en el ámbito de la ciberinteligencia un futuro laboral[3], encontrando una fuerte demanda dentro del sector de la seguridad y defensa[4] .

Cabe destacar también cómo hemos visto en los últimos meses una reivindicación ciudadana, que se ha visto reflejada en una nueva regulación europea[5] ,ha hecho replantear los modelos de negocio de los mayores gigantes tecnológicos como se ha visto recientemente con el caso de

[1] "Scientia potentia est", Sir Francis Bacon

[2] Artículo 11 de "la Carta de Derechos Fundamentales de la Unión Europea". http://www.europarl.europa.eu/charter/pdf/text_es.pdf [21 de junio, 2018]

[3] "The Institute: The Cybersecurity Talent Shortage Is Here, and It's a Big Threat to Companies". https://cybersecurity.ieee.org/blog/2017/04/13/the-institute-the-cybersecurity-talent-shortage-is-here-and-its-a-big-threat-to-companies/ [21 de junio, 2018]

[4] "El ciberespacio y la accesibilidad de la tecnología han rebajado el coste de la información y ampliado su acceso, pero agravado su vulnerabilidad", en Estrategia de Seguridad Nacional 2017. http://www.dsn.gob.es/es/estrategias-publicaciones/estrategias/estrategia-seguridad-nacional-2017 [20 de junio de 2018]

[5] Regulación General de Protección de Datos. https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:32016R0679 [21 de junio, 2018]

Cambridege Analytica[6]: el derecho a la privacidad y a la protección de datos . Hace cuestión de poco tiempo estos gigantes se aprovechaban del desconocimiento general y de la falta de legislación al respecto y recogían toda la información que nosotros como usuarios generábamos directamente(y sabíamos, o no, que ellos registraban), así como toda la información creada de manera indirecta que íbamos dejando al usar cualquier tipo de aparato tecnológico en nuestro día a día. El estudio de maneras con las que protegernos de este tipo de abuso se sale del ámbito de este trabajo, pero se tiene presente en todo momento la idea de no abusar y de mantener siempre la legalidad[9].

La inteligencia en el sector en que nos encontramos puede entenderse como el producto obtenido de recolección, evaluación, análisis e interpretación de información en bruto recaba de Internet [7]. A día de hoy si hablamos de ciberinteligencia se suele pensar en el proceso o servicio por el cual se analiza e interpreta información respecto a amenazas con el fin de disminuir los riesgos de sufrir ataques informáticos. No obstante, aunque este sector está en auge, la ciberinteligencia no se limita a evitar cibercrímenes. La creación de conocimiento es el fin que se busca y en este ámbito las aplicaciones son infinitas. Desde conocer a la clientela de tu negocio online, estar al tanto de acontecimientos globalmente relevantes, conocer las modas que van surgiendo o tener vigilados los productos que se comercializan en las redes oscuras"; son todos ejemplos de variantes de aplicaciones de ciberinteligencia.

Aunque las posibilidades son muchas, todas las aplicaciones de ciberinteligencia van a tener un esqueleto de funcionamiento común. En primer lugar, cualquier sistema que quiera extraer información de la red y transformarla en conocimientos útiles y relevantes va a necesitar una primera fase de enfoque. Será necesario estudiar de dónde se pretende recoger la información, y qué tipo de información se busca (los productos que ofrece la web de un competidor, los gustos y aficiones de los clientes de tu producto, opiniones contrarias a tu partido político en foros clandestinos, etc.). Una vez se tenga el enfoque claro comienza la fase de extracción de información. La manera con la que vayamos explorar estas webs con el fin de recuperar información va a depender mucho según el ecosistema en el que nos movamos. No va a ser tan fácil recoger información en sitios alojados en redes profundas, en las que los usuarios son casi anónimos y en las que para acceder a contenidos de tu interés es necesario interactuar con la pagina de nuevas formas, y no como lo podría ser antes, leyendo información de una web alojada en Internet en la que todo el contenido se comprende en un archivo HTML plano.

El proceso de creación de inteligencia tendrá fases posteriores a esta, no obstante, este trabajo se va a centrar en justamente la extracción de contenido perteneciente a la *Deep Web*. Los principales retos que se encuentran en esta temática serán la automatización de herramientas que puedan extraer grandes cantidades de información. Más adelante se explicará mejor qué son y el funcionamiento de este tipo de redes, en las cuales la extracción de datos cobra una complejidad importante.

Por último, considero importante volver a recalcar que este trabajo pretende, en todo momento, tener en cuenta el marco legal y ético. Tanto el hecho de estudiar cómo recabar información de la web y de aprender sobre redes oscuras, así como el interés en perseguir una carrera profesional cercana a la ciberinteligencia ha sido lo que ha motivado la elección de este tema como trabajo de investigación.

[6] Confessore, Nicholas. "Cambridge Analytica and Facebook: The Scandal and the Fallout So Far" https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html [18 de junio, 2018]

[7] "Las Fases de la Ciberinteligencia" . http://blog.elevenpaths.com/2016/03/las-fases-de-la-ciberinteligencia.html [21 de junio, 2018]

1.2. Objetivos y enfoque

Este trabajo persigue conseguir varios objetivos. En primer lugar se pretende aprender sobre qué es la *DeepWeb*, qué es la *DarkNet* , en qué se diferencian de la *Surface Web* y qué hace que sea difícil indexarlas y extraer contenido de ellas. Para poder entender bien las dificultades, se estudiará en profundidad qué es un indexador. Se va a definir bien su funcionamiento y se hablará de casos prácticos funcionales en Internet.

En segundo lugar, se estudiará cómo indexar la Deep Web. Se pretende definir las diferentes maneras de poder hacerlo y estudiar casos prácticos en varios ámbitos: veremos indexadores para Deep Web e indexadores específicos de Tor que llevarán ciertas dificultades añadidas al proceso de recogida de información. Se ha elegido estudiar los *servicios ocultos* pertenecientes a Tor por varias razones: componen es la red oscura con más usuarios del mundo y por consiguiente la que más contenido alberga. Su uso indebido por entidades y personas que la usan como mecanismos de esquivar la ley inspira en mi un interés en estudiar esta tecnología y ver cómo poder crear aplicaciones con las cuales si se podría asegurar el cumplimiento con la ley en estos entornos.

Finalmente, este trabajo procurará marcar un itinerario a seguir para poder desarrollar un indexador propio en el futuro, así como adentrarse un poco en alguno de los ámbitos de trabajo del dominio multidisciplinar que componen la ciberinteligencia.

1.3. Metodología y plan de trabajo

Este trabajo se va a estructurar de la siguiente manera:

- Capítulo 2. Estado del arte: este capítulo definirá todos los conceptos con los que trataremos durante el trabajo. Se explicará las diferencias entre Surface Web, DeepWeb y DarkNet. Se explicará qué es un indexador, de qué se compone y cómo funciona.
- Capítulo 3. Indexar la DeepWeb: se estudiarán las diferencias en la indexación de contenido alojado en la DeepWeb. Se explicará en profundidad los mecanismos de recuperación de información posibles y se verán casos prácticos.
- Capítulo 4. Conclusiones y trabajo futuro.

2

Estado del arte

2.1. La DeepWeb

Para que puedas realizar una búsqueda en Google sus rastreadores web han realizado un proceso de recoger información de cientos de miles de millones de páginas web [10]. Aunque este número pueda parecer que englobe a un porcentaje considerablemente grande de toda la web, recientemente se ha comenzado a ver no es más que la "punta del iceberg". Debajo de esos miles de millones de webs a las que podemos acceder consultando a través de buscadores hay un océano inmenso de webs a las que no tenemos este mismo acceso directo y esto se debe a que el mecanismo para acceder a la información que esté alojado en ellas, o al que podamos acceder a través de ellas, es diferente al mecanismo de recogida de información del internet que conocemos, la *Surface Web*. A todo este conjunto de páginas y bases de datos a cuya información no tenemos acceso de una manera tan directa se conoce como *Deep Web*.

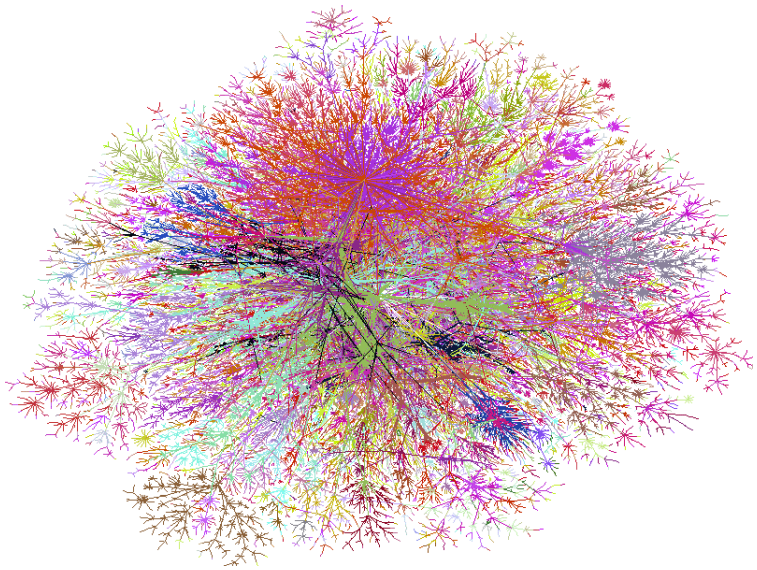

Figura 1: Representación gráfica de internet por Hal Burch y Bill Cheswick en "Mapping the Internet" [1]

Hay una concepción errónea sobre qué es la *Deep Web*, confundida muy a menudo con la *Dark Net*. Las *Dark Nets*, o redes oscuras, son solo una parte pequeña dentro de todo el conjunto de de la *Deep Web*, con características específicas (que se explicarán más adelante) que las han hecho populares para usos delictivos o éticamente cuestionables. No obstante, hay que entender que, aunque las redes oscuras se encuentran contenidas dentro de la *Deep Web*, esta no es solamente eso, si no que engloba todo un conjunto de contenido al que no hay acceso de manera tan sencilla como una consulta a Google, pero que es perfectamente legal y justificable. Se va a proceder, a continuación, a explicar qué es la *Deep Web*, en qué se diferencia de la web de superficie que conocemos y qué tipo de contenido podemos encontrar.

En sus orígenes, la web estaba compuesta por unas pocas páginas estáticas con una estructura HTML sencilla y era posible mantener un directorio central en el que poder englobar la mayoría del contenido de Internet. Con el crecimiento de la popularidad de internet y la aparición de cientos de miles de nuevas páginas web diariamente, se vuelve inviable tener un directorio central con todo el contenido publicado en la red[11]. Aparecen buscadores web, los cuales comienzan a rastrear internet con el fin de poder crear indices con los que dar posibles resultados a peticiones de usuarios. Internet se convierte en la red que hoy en día conocemos, en la cual para que una página web sea encontrada por los rastreadores de los buscadores debe estar referenciada a través de enlaces de hipertexto por otras páginas. De hecho, cuantos más enlaces hagan referencia a una página web más fácil será de encontrar y mayor será la puntuación de popularidad en indices de buscadores como Google, gracias a la cual está página estará posicionada más arriba en la lista de resultados devueltos al usuario que haga una consulta.

No obstante, en los años 90 aparece un fenómeno que hace que el rastreo de contenido en internet cambie y se deje de poder tener acceso a una gran y creciente parte de la web. Se introducen tecnologías de bases de datos a internet y pronto se convierte en la tendencia del desarrollo web. Páginas webs que antes tenían miles de documentos estáticos en páginas HTML fijas ahora comienzan a usar tecnologías de generación dinámica como resultado de consultas a sus bases de datos. El acceso a toda la información contenida en estas bases de datos no es el mismo tipo de acceso que se requería para webs estáticas[11]. A día de hoy hay miles de ejemplos de páginas web cuyo contenido está alojado en bases de datos y por consiguiente está öcultoä los buscadores. Cuando se habla de *Deep Web* se hace muy habitualmente referencia a material científico, médico, gubernamental, legal, al cual para poder llegar se tiene que interactuar con formularios web, tener perfiles de usuarios con permisos de entrada o simplemente hacer consultas a un buscador propio[12]. Toda la información en webs de compra-venta de casas, o de aerolíneas, un vídeo publicado en YouTube de manera privada, la intranet de la empresa en la que trabajas, tu cuenta en la web de tu banco con todos tus datos de operaciones o tu perfil en LinkedIn son todos ejemplos de contenido que se encuentra en la *Deep Web* (figura 2).

Pese a lo que pueda parecer, la linea divisoria entre la red de superficie y la red profunda no es una linea clara, y hay numerosos casos en los que contenido de una de las dos partes pueda aparecer en la otra. Más comúnmente se da la situación en la cual información contenida en la *Deep Web* sale a flote. Podemos tomar un ejemplo en el cual se realiza una búsqueda de cierto producto en Amazon, visto en la Figura 3. Si nos fijamos, la URL que devuelve la consulta contiene, entre otras cosas, las palabras que hemos introducido en la barra de búsqueda. Si elegimos un producto de los mostrados también veremos una URL que contiene tanto la información del producto como de nuestra propia búsqueda realizada (figura 4). Si yo ahora decidiese dicho enlace publicarlo en alguna página web estática como un blog o foro, dicho enlace podría ser rastreado por buscadores y aparecerle a alguien en una futura consulta a Google, Yahoo, etc.

Podemos ver que no obstante esto no siempre va a ser el caso en webs con contenido almacenado en bases de datos. Podemos ver en la siguiente captura de la página de Ryanair como no se genera ninguna URL que pudiésemos reutilizar para recibir los mismos resultados (figura

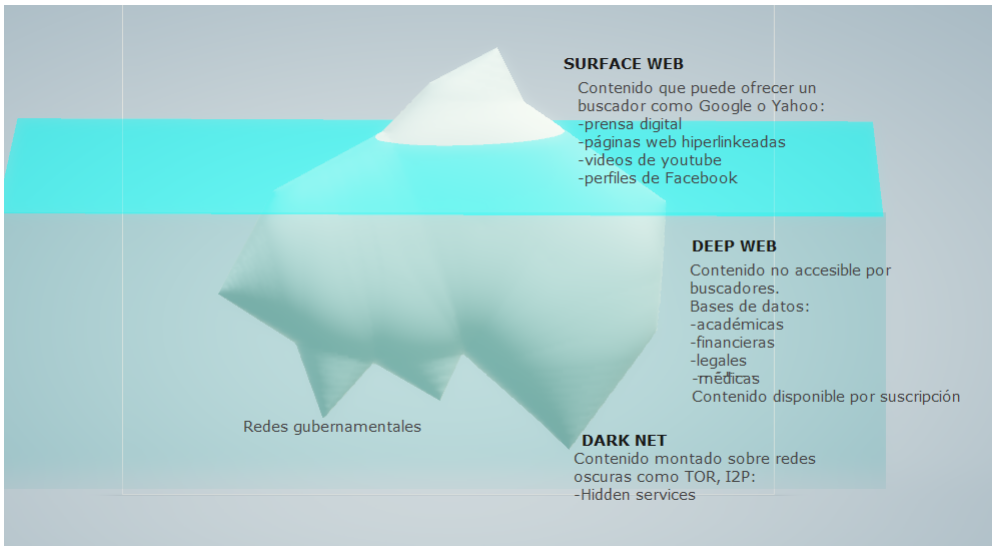

Figura 2: Gráfico de representación de la web

Figura 3: Captura de una búsqueda en Amazon.es

5).

Aunque hay numerosos artículos y estudios que estiman el tamaño de la *Deep Web*, llegando a decir que constituye unos porcentajes altísimos del contenido de toda la web, en la práctica es imposible poder medir con el más mínimo criterio de fiabilidad el tamaño de la *Deep Web*. Se ha de tener en cuenta que, aunque hay webs que reconocemos como pertenecientes a la *Deep Web*, hay muchas más que no se conocen y que no están localizadas. No solo eso, sino que tampoco se puede estimar el contenido albergado en dichas webs. Intranets estatales y privadas pueden llegar a tener dimensiones muy superiores a lo que imaginamos. También resulta muy complicado poder estimar el número de redes privadas más profundas aún, las conocidas como *DarkNets*, que se soportan con grandes medidas de seguridad. En estas redes es muy típico que webs alojadas en ellas estén publicadas en intervalos específicos de tiempo, por lo que puede darse el caso en el que durante largos periodos de tiempo el contenido no sea accesible al usuario.

Figura 4: Captura de la ficha de un producto accedido a través de la búsqueda mostrada en la Figura 3

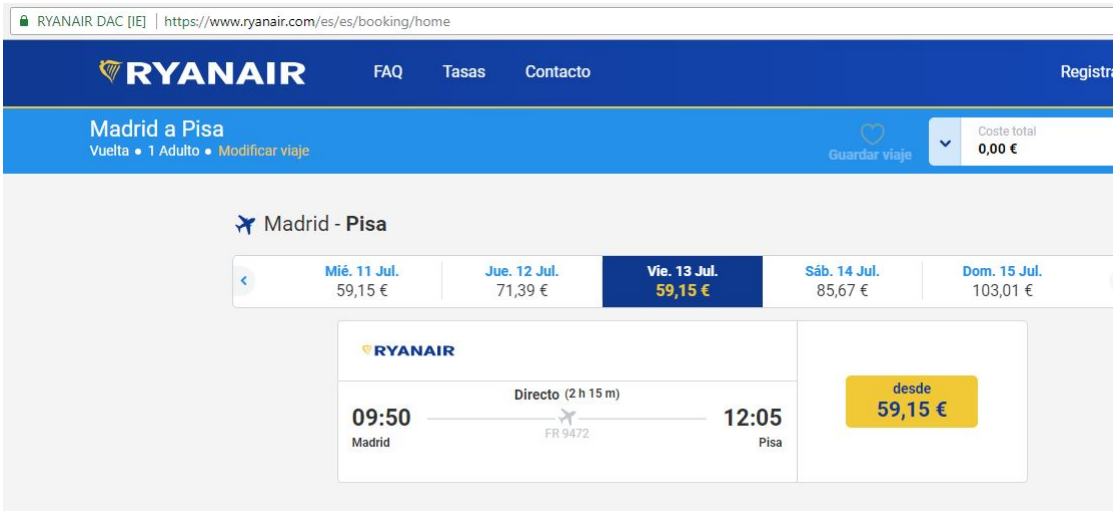

Figura 5: Captura de una búsqueda realizada en Ryanair.com

Tipos de anonimato

Es habitual relacionar la "profundidad"de la web con las medidas de seguridad, privacidad y anonimato con las que se sustentan las webs alojadas en dichos niveles. Los términos de privacidad y anonimato se vuelto características casi de moda en la sociedad. El debate generado en los últimos años sobre el abuso de grandes empresas sobre nuestro desconocimiento ha hecho que ahora cada individuo se convierta en un agente activo en la lucha contra el mantenimiento de la privacidad en la red. Se ha sumado a este fenómeno la reciente Regulación General de Protección de Datos[1], cuyo objetivo principal es la protección de los individuos en lo que respecta a sus datos personales.

La privacidad en internet puede entenderse de diversas maneras. Primero, al hablar de privacidad se puede entender como confidencialidad: un servicio web se compromete a mantener tu privacidad cuando minimiza o asegura que tus datos personales recolectados serán mantenidos confidencialmente. Otro entendimiento es decir que el usuario es dueño de su privacidad, por lo que él puede decidir si quiere ocultar información personal o si quiere difundirla. Por último,

[1]Regulación General de Protección de Datos. https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:32016R0679 [21 de junio, 2018]

también se podrá entender la privacidad como una práctica llevada a cabo por los diferentes mecanismos que facilitan al usuario la intervención a la hora de decidir si compartir o no sus datos personales [13].

El anonimato en la web no es lo mismo que la privacidad. Por anonimato se entiende el hecho de que no se pueda identificar un perfil digital con una persona física. Hay diversos mecanismos que ayudan a romper la conexión entre estas dos identidades. Al interactuar con la web nos encontramos con los siguientes tipos de anonimato:

1. Anonimato de emisor: es uno de los mecanismos más practicados. En este caso, el usuario es el origen de una comunicación y se pretende que no pueda ser identificado. Se suele procurar evitar dejar rastros al usar la web, así como ocultar cualquier información personal sobre su persona, de manera que ningún observador pueda identificarle.

2. Anonimato de receptor: no solo es relevante en un ecosistema poder defender al usuario que vaya a consultar información, también lo es que un usuario pueda publicar contenido de manera anónima. Poder proveer servicios webs sin revelar su ubicación ni identidad pone una barrera para posibles atacantes [14].

3. Anonimato de la comunicación: este tipo de anonimato será el que persiga que no se pueda vincular la comunicación entre un emisor y un receptor.

Es relevante también mencionar que un aspecto clave para el mantenimiento de la privacidad y el anonimato recae fuertemente sobre la estructura de las webs con las que se interactua. Dará igual que se tomen numerosas prácticas con el fin de preservar tu identidad confidencial si luego al hacer un envío de información, dicho contenido no va encriptado y los datos van expuestos para que cualquier observador los recoja[2].

2.2. La DarkNet

Las *Dark Nets* son un conjunto de redes que generalmente están protegidas por VPNs para los cuales se necesitan clientes específicos para entrar. La mayoría del contenido que se encuentra en estas redes pertenece a páginas que no van a poder ser indexables por buscadores convencionales y esto se debe a varias razones. En primer lugar, las redes oscuras más grandes (el tamaño se define en numero de usuarios activos) proporcionan un sistema de resolución de dominios web diferente al que se usa en la web de la superficie, por lo que no se puede acceder a dichas páginas a través de navegadores normales. En segundo lugar, estas redes suelen tener un gran carácter voluntario y en numerosas veces la propia red esta conformada por *relays* cedidos voluntariamente por los usuarios. Esto conlleva que pueda haber mucha información que no se encuentre accesible o publicada en todo momento.

Estas redes son las notorias por ser asociadas con los núcleos de cibercrimen y de mercados negros. Aunque a día de hoy haya una fuerte superioridad de contenido ilícito en estos tipos de redes, cabe destacar el origen de las mismas. En su estructura, las *Dark Nets* proporcionan al usuario tanto anonimato de emisión como de recepción. Estas propiedades configuraron una red a través de la cual ciudadanos que viviesen en países con altos grados de censura pudiesen tener acceso a información prohibida y pudiesen ejercer su derecho a la libertad de expresión. Esto era posible ya que el usuario queda protegido por la red y la aplicación de la ley no llega. Este tipo de redes son en sus totalidad herramientas neutrales que pueden ser usadas para bien o para mal. Desafortunadamente nos encontramos con que la tendencia muestra que en países

[2] "Plaintext over Tor is still plaintext". https://blog.torproject.org/plaintext-over-tor-still-plaintext [21 de junio, 2018]

liberales en los que presumimos de derechos y libertades son en los que más se tienden a usar con propósitos malintencionados [15].

2.2.1. Tor: The Onion Router

Tor es la red anónima más concurrida del mundo. Se estima que unos 2 millones de usuarios la usan a diario, y que hay aproximadamente unos 30.000 *relays* voluntarios conectados a la red, cediendo ancho de banda para su mejor funcionamiento.

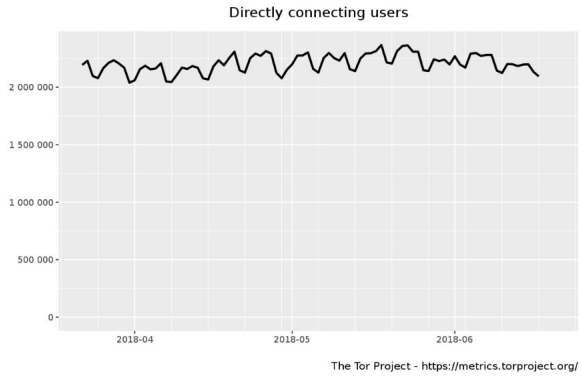

Figura 6: Estimación de usuarios en la red Tor analizando peticiones inducidas por clientes a puentes y nodos. Captura de [2]

Tor es un software libre y compone una red abierta que ayuda al usuario a defenderse ante análisis de tráfico web, un tipo de vigilancia de la red que amenaza su libertad personal y privacidad, la confidencialidad de sus actividades de negocios y amistades, y la seguridad del estado. [3] La manera en la que esta red consigue proteger la privacidad de los usuarios se basa en proteger el origen de la información en todo momento. Cuando el usuario consulte una página web, su información será rebotada entre diversos nodos de la red y en cada uno de ellos se añadirá una capa de cifrado terminando finalmente como una cebolla. De esta manera se evita que si un observador interceptase el paquete pudiese descifrarlo e identificar a su emisor.

Los principales mecanismos con los que Tor asegura el anonimato de emisor, así como de receptor, son los siguientes. En primer lugar, con el fin de ocultar el origen de las comunicaciones, cuando un cliente se conecta a un servidor su tráfico es enrutado a través de un circuito de tres nodos de la red. Estos nodos pueden ser o elegidos por el cliente o de manera aleatoria y con cada conexión nueva se construirá un nuevo circuito. El cliente utiliza mecanismos de negociación de claves publicas con los nodos de su circuito. Para enviar la información, el cliente genera un paquete y lo encripta iteradamente con las tres claves de sesión de los nodos de su circuito. Conforme en paquete se mueve a través del circuito, cada nodo quita la última capa de encriptación con su clave privada, y la información desencriptada solo expondrá a que nodo se ha de reenviar el paquete. De esta manera, cada nodo solo conoce la procedencia del paquete desde el nodo estrictamente anterior a si mismo y el destino estrictamente siguiente al suyo. El servidor final recibirá el paquete original pero no conocerá tampoco desde dónde fue enviado.

La manera con la que se protege la privacidad del receptor es a través de los conocidos *hidden services*. Los *hidden services* son las páginas y servicios webs que conocemos en la red de superficie, pero en Tor y con algunas peculiaridades. En primer lugar, estos servicios tendrán que estar basados en el protocolo TCP para poder montarlos sobre la red. En segundo lugar, la

[3] "What is Tor¿' https://www.torproject.org/ [21 de junio, 2018]

manera de hacerse publico es muy diferente a como se hará en la red de superficie. En Tor no existen mecanismos centralizados para la resolución de dominios, es decir, no existen servidores DNS. En su lugar, encontramos repositorios denominados *Hidden Service Directories*. Cuando un cliente quiera publicar su servicio deberá seguir los siguientes pasos:

1. El cliente deberá configurar su Onion Proxy, el cual escogerá un pequeño numero de *relays* como *Introduction Points* y establecerá circuitos de introducción a cada uno de ellos.
2. El cliente generará un descriptor, que contendrá la lista de *Introduction Points* así como una clave publica de encriptación, y un ID para su Hidden Service, y lo publicará en algunos HSDirs.

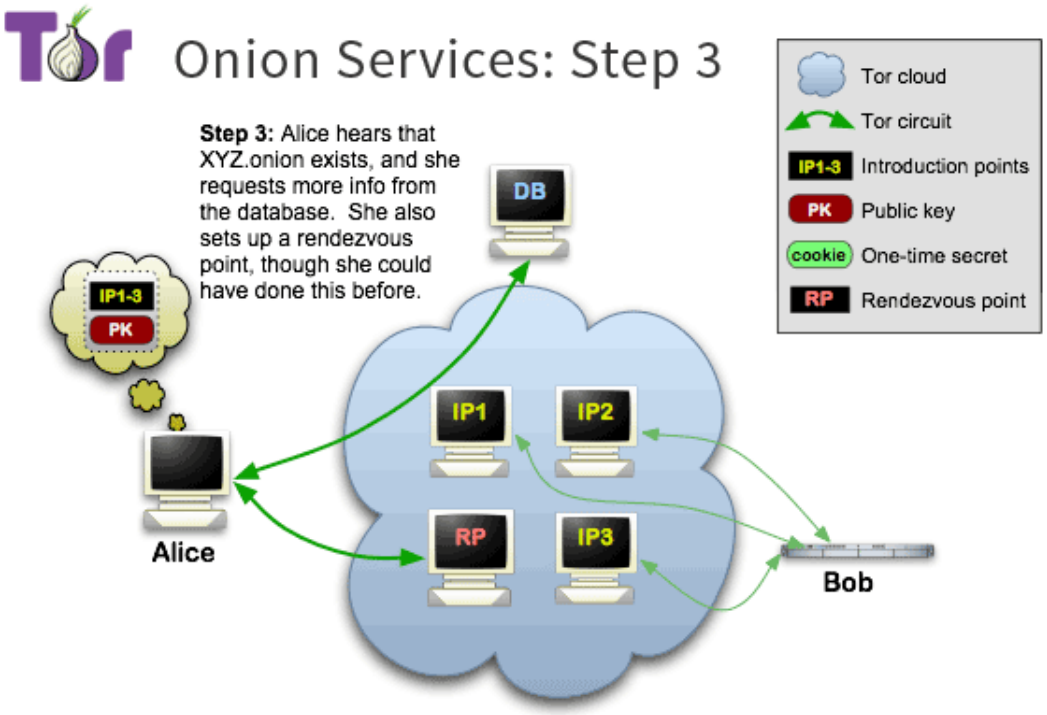

Figura 7: Esquema de comunicación entre dos usuarios en Tor extraída de https://www.torproject.org/docs/onion-services

Cuando otro cliente quiera conectarse al servicio publicado deberá primero haber conseguido de alguna manera externa el dominio .onion del HS. Luego deberá consultar a los HS Directories para hacerse con el descriptor del servicio y conocer así los *Introduction Points* al mismo (figura 7). Para asegurar el anonimato de comunicación, en vez de conectarse directamente al *Introduction Point*, el cliente escogerá un nodo de la red aleatoriamente y montará un protocolo de *Rendezvous*. El cliente enviará un paquete con los datos del *Rendezvous*, encriptados con la clave publica del HS, al punto de introducción, a través de un nuevo circuito. EL *Introduction Point* enviará esta información al Hidden Service a través de su propio circuito. Cuando el cliente que ha publico el servicio reciba la invitación de *Rendezvous* la podrá desencriptar con su clave privad ay construir un circuito al nodo de encuentro. A partir de ese momento, el nodo de *Rendezvous* será el encargado de distribuir los mensajes de cliente a cliente a través de sus circuitos. De esta manera se ha construido un camino de comunicación en el cual el cliente no conoce la dirección ni el dueño del servidor, y el servidor no conoce la dirección ni al dueño del cliente.

Aunque existen numerosas redes anónimas aparte de Tor, como I2P o FreeNet, etc., este trabajo se centra en la indexación de contenido de la *Deep Web* externa a estas redes. No obstante, se ha considerado que Tor es relevante para esta investigación. Al ser la red oscura más usada se puede asumir que también es la que más contenido alberga. Es por tanto una parte de la red que no se puede indexar con metodologías tradicionales pero que constituye uno de los focos de interés más grandes para las nuevas técnicas de indexación por el potencial valor que se podrá extraer si se consigue. Hay numerosas empresas que han empezado a ver que la ciberinteligencia enfocada a estas redes va a constituir un nuevo mercado y negocio, que no ha sido explotado aun.

2.3. ¿Qué es un indexador?

Internet, o la World Wide Web, es un sistema que se estructura como un grafo en el que cada página web se corresponderá a un nodo y los hiperenlaces serán los enlaces entre nodos. Es por tanto, que el proceso de rastreo llevado a cabo por un indexador puede entenderse como un problema de búsqueda en un grafo. En este capitulo vamos a explicar detalladamente qué es un indexador y cómo se realiza el proceso de rastreo o de çrawl"(en inglés). En primer lugar, un indexador es un programa que se mueve a través la red y busca acceder al mayor numero posible de páginas webs diferentes y almacenarlas o algún tipo de información que las represente. Generalmente se relaciona el concepto de indexador con el de un buscador web, aunque no siempre tiene porqué ser esta la única finalidad de rastrear la web. Es posible que entidades o particulares quieran mantener una base de datos propia con el fin de encontrar información especifica relevante a sus intereses. No obstante, los buscadores webs a día de hoy hacen de puertas de entrada a la web por lo que su relevancia en el contexto de indexación de contenido es sumamente considerable. A los los buscadores web les va a interesar conseguir la máxima cobertura de la web posible, así como mantener su información actualizada.

Por continuar el símil del grafo, el problema de rastrear la web comenzará con una lista abierta de nodos "semilla". El indexador se moverá de nodo en nodo a través de los enlaces que encuentre al explorar cada nodo o página web. Un indexador podrá ser selectivo, caso en el cual intentará solo seguir enlaces que crea que le llevarán a nodos que contengan información especifica del formato o temática que este explorando. Según el tipo de búsqueda escogido, se podrá emplear una búsqueda en anchura "*breadth-first*.º una búsqueda según prioridades "*best-first*". Para esta segunda será necesario puntuar los enlaces encontrados en cada nodo según el contexto para decidir cuál recorrer primero. En este capítulo vamos a explicar cómo funciona un indexador de web normal. Se va a detallar el proceso de rastreo convencional para páginas webs estáticas o parcialmente estáticas, deteniéndonos levemente en cómo se extrae información de ellas. Seguidamente se procederá a explicar cómo cambia el enfoque cuando se pretende rastrear la *Deep Web*, qué partes del proceso necesitan una adaptación al nuevo entorno y cómo se consigue.

2.3.1. Estructura de un indexador

Cuando se habla de indexadores en general se tiene a pensar en buscadores webs. Aunque ya hemos comentado que las motivaciones de este trabajo no se centrar en crear un buscador, sí es relevante entender como los indexadores, y en su mayoría los buscadores webs, están estructurados. Si no atenemos a lo que la palabra indexador se refiere, el primer componente en la estructura es obvio: un índice. Los índices tienen una función clara, poder ordenar información de manera que sea accesible a posteriori.

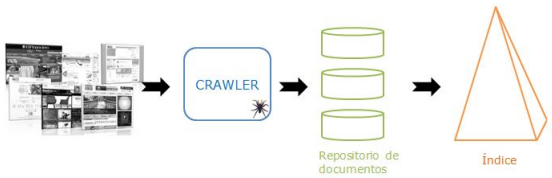

Figura 8: Estructura simplificada de un indexador

El proceso de indexar comienza por la elección de una estructura de datos en la cual se guardarán los documentos e información. Esto no será tarea fácil: la web no tiene una estructura definida por lo que la información descargada será completamente diferente una de otra. Se deberá encontrar un formato general al que poder convertir todos los datos que se vayan a descargar. Estos datos no estarán constituidos únicamente por el contenido de las webs, si no que habrá que recoger también metadatos que puedan brindar información importante. Según el alcance propuesto, también habrá que tener en cuenta las dimensiones del contenido que se pretenderá guardar, por lo que será necesario usar técnicas de compresión de datos.

La siguiente fase de la indexación consistirá en comenzar la extracción de las palabras clave, o *keywords*, de los documentos descargados. Generalmente se buscará información relevante en los títulos, subtítulos o en lugares del documento que se usen para definir su contenido.Muy posiblemente en esta fase se deban unir *parseadores* de lenguaje natural que podrán ayudar en el proceso de *tokenización* (convertir un texto en unidades independientes representativas del mismo), en el *stemming* (identificar que dos palabras pertenecen a un mismo grupo semántico) y en el *stopwording*(quitar palabras que no aportan significado). Terminada esta fase, faltará construir el índice. Se llevará a cabo un proceso de pesado de los términos o *tokens* según frecuencias estadísticas u otros criterios definidos. El tipo de índice que se construya dependerá de las prioridades del proceso. Se tendrá el cuenta la complejidad de la búsqueda y el tamaño del índice y se escogerán formatos de datos diferentes según se premie más una cosa que otra. Generalmente, los indices se construirán con alguna de las siguientes estructuras:

1. Índice invertido: probablemente la estructura más usada en el contexto de indexación de contenido web. Los índices invertidos mapean palabras o frases a documentos que estén guardados en repositorios o bases de datos. Generalmente se presentarán en formatos de tablas hash o árboles binarios.

2. *Suffix tree o suffix array*: son estructuras de datos que guardan los sufijos de las palabras que indexan. Son populares por proporcionar búsqueda de strings con complejidad lineal. Aunque la construcción es algo más costosa, otras funcionalidades las hacen muy atractivas, por ejemplo, en aplicaciones para análisis de genomas[16].

3. Matriz bidimensional: para las propuestas menos exigentes se podrá limitar a guardar en una matriz las ocurrencias de palabras en documentos.

En último lugar, aunque será la parte con la que comience todo el proceso de indexar, estará el **rastreador o "*crawler*"**. Este elemento es en el que más hincapié se va a hacer en este trabajo porque es el que más interés despierta y más importancia recaba cuando nos llevamos el indexador al contexto de la *Deep Web*. El crawler de lo que se va a encargar va a ser no solo de recorrer la web y encontrar nuevas páginas a las que visitar, si no de interactuar con las webs. Esto último constituirá el mayor reto a la hora de indexar la *Deep Web*. La figura **XXX** muestra la estructura y funcionamiento básico de un crawler, el cual va a ser explicado en mayor detalle en el siguiente apartado.

2.3.2. Estructura de un crawler

El funcionamiento de un crawler va a consistir en un ciclo, al cual llamaremos bucle de rastreo, que se va a repetir hasta que alguna condición de fin se cumpla. El bucle va a consistir en los siguientes pasos:

1. Coger de la lista de URLs no visitadas la que se encuentre en primera posición.
2. Recuperar el contenido de la página web correspondiente a la URL escogida.
3. Parsear la información obtenida, buscar enlaces a otras páginas webs.
4. Añadir enlaces encontrados a la lista de no visitados y añadir la página recién visitada al historial y el contenido a un repositorio o índice.

El proceso puede terminarse si se llegase a un punto en el que la lista de URLS no visitadas estuviese vacía, aunque esta situación no suele ser muy común. También se podrá detener el proceso si se consigue visitar un numero establecido de páginas o si se finaliza un objetivo predefinido relacionado con información relevante recogida, etc.

Frontera

Como se ve en la figura **XXX**, la frontera es el componente desde el que se parte para comenzar a rastrear. La frontera contendrá la lista de la cual se van a ir extrayendo URLs de páginas webs no visitadas. Según el rastreador, su estructura y su alcance, se escogerá diferentes opciones para implementar la frontera. Principalmente variará dónde se guarda la información, si en memoria o en disco, y en qué estructura de datos. Según la capacidad de nuestros sistemas y las prioridades que tengamos, si preferimos velocidad o nos prima más limitar nuestro consumo de memoria. Las estructuras de datos entra las que generalmente se escogerá para constituir la frontera son una cola *FIFO* (*First-in-first-out*) o una lista de prioridad. La primera opción consiste en una lista en la que el elemento que se extraiga va a ser el que se encuentre en primera posición en la lista, y cada elemento que se vaya a añadir irá al final de la lista. Esta opción nos conformaría una búsqueda en anchura, lo cual, como veremos más adelante no va a ser la medida más usada.

La segunda opción es una variante de esta primera, una lista de ordenada según prioridad. En esta implementación la URL que se vaya a extraer será aquella con mayor puntuación de la lista, que se corresponderá con la primera, y las URLs que se vayan a insertar tendrán que haber sido puntuadas con algún algoritmo de heurística que tendrá que haberse desarrollado y se colocarán en la posición que corresponda a su puntuación. De esta manera tendremos una búsqueda preferente.

Para ambas opciones tendremos un problema y es que ambas estructuras no proporcionan un rendimiento bueno a la hora de buscar si un enlace que queremos insertar ya existe en nuestra lista, ya que la búsqueda lineal no nos va a proporcionar los mejores resultados. Para la primera de las opciones descritas no hay ninguna otra manera de buscar un enlace ya existente menos que recorriendo la lista entera; y para la segunda opción hay que tener en cuenta que un mismo enlace probablemente sea referenciado desde diferentes páginas por lo que no se puede asumir que una URL al pasar por el algoritmo de heurística de puntuación siempre será asignado el mismo score.

La solución casi siempre propuesta e implementada en la literatura [17] es recurrir a una tabla hash con bajas colisiones mantenida en memoria aparte de la lista de la frontera, a la cual se acceda con los URLs como claves. Esta tabla tendrá que estar sincronizada en todo

momento con la lista de URLs. También existiría la posibilidad de convertir toda la lista de nodos no visitados en una tabla hash. Una vez más, tendría que buscarse un algoritmo hash con mínimas colisiones y aunque en este caso la búsqueda de si existe ya un enlace en la tabla ofrece un buen rendimiento, no sucede lo mismo a la hora de extraer la siguiente URL que se quiera visitar. Dependiendo del mecanismo previo elegido, la tabla contendría como claves el resultado hash de las URLs por visitar y como contenido podría contener la propia URL así como un timestamp (si se está implementando una búsqueda en anchura) o una puntuación (si se trabaja con búsqueda "best-first"). En ambos casos, el rendimiento de búsqueda no ofrece los mejores resultados.

Según la estructura que se elija también cambiará el comportamiento de nuestro rastreador cuando la frontera esté llena. Con una lista *FIFO*, se podrá insertar solo el primer enlace encontrado en el bucle en que se encuentre, mientras que con una lista de prioridad se elegirá el enlace con mayor puntuación.

Historial y repositorio

De igual modo que se busca en la lista de nodos no visitados para no incluir enlaces que ya estén en la lista, también se tendrá que buscar en una lista que contenga el historial de páginas visitadas para evitar así revisitar páginas que ya se conocen. No solo con este fin se podrá usar el historial, si no que también se podrá ver el camino seguido por el rastreador con el fines analíticos y para mejorar los algoritmos de búsqueda. El historial estará compuesto generalmente por un diccionario que guarde las URLs como claves y un *timestamp* de cuándo fueron exploradas. El timestamp también podrá ser útil para decidir si revisitar la página por si pudiese no estar actualizada. Habitualmente el historial estará almacenado en memoria para tener un acceso rápido. Normalmente también se trabajará con un repositorio en el que se almacenará la información extraída de las páginas webs exploradas. La información será archivada en documentos independientes por cada página web visitada en los que el nombre de cada archivo se corresponderá con algún código hash de la URL o con el dominio base de la página web, etc. Según el alcance y tamaño del rastreador con el que se trabaje, también se podrá usar mismamente el respositorio de URLs como historial, de manera que si se quieres buscar si algun enlace ha sido previamente visitado se deberá consultar si existe un documento que se corresponda a dicha página web. Esta consulta puede resultar más lenta, pero si se puede comprometer algo de velocidad por memoria podría ser una opción a considerar.

Recuperación de información

El siguiente componente será el encargado de conseguir la información de las páginas webs que se quieran visitar. Será necesario levantar un cliente HTTP que pueda realizar consultas y recibir y entender las posibles respuestas. Tiene especial importancia la segunda parte, el poder entender los "HTTP responsesza que en muchos casos al intentar visitar una página, esta nos redireccionará a otra. Esta casuística nos obliga a tener que poder entender y parsear las cabeceras HTTP de respuestas.

El *parseado* de las webs descargadas y la extracción de enlaces correspondería al siguiente punto que se va a explicar. No obstante, con el fin de optimizar el proceso de recogida y descarga de páginas webs a menudo se aplica la técnica de "*prefetching*". Muy a menudo, el mayor tiempo invertido en el proceso de recuperación no pertenece a la transferencia del protocolo HTTP, si no a la resolución del dominio web [18]. El *prefetching* intentará disminuir esta latencia de la siguiente manera: cuando una página web haya sido extraída se intentará construir un flujo de etiquetas HREF y se enviarán peticiones DNS de resolución de dominios. Esto se hará antes de haber llevado a cabo el proceso de *parseo* y reconstrucción de HTML, además de otros

procesos que se explican a continuación. De esta manera se conseguirá evitar posibles retardos y momentos de bloqueo por esperar estas resoluciones de direcciones, teniendo ya parte de las URLs encontradas resueltas.

Por ultimo, también es relevante comentar el protocolo *robots.txt*, que dicta un mecanismo u convención por la cual se establece si un servidor no quiere que ciertas páginas sean rastreadas por crawlers o robots. Las páginas webs publicarán este documento generalmente en el archivo raíz de la página web y especificará una lista de rutas y subdominios que el crawler no estará autorizado a rastrear. En estos archivos también se podrá especificar que usuarios o agentes serán los que tendrán o no permisos, ya que generalmente se suelen hacer distinciones entre robots rastreadores de buscadores webs del resto. Este fichero, no obstante, puede crear problemas de privacidad ya que será lo primero que consulte un usuario que busque información confidencial o que se pretenda ocultar. Será recomendable siempre no publicar información privada en páginas webs sin algún tipo de filtro adicional de seguridad.

Parseador y extractor de nuevos enlaces

Por último lugar, tras la recuperación de una página web se procederá a extraer la información contenida en ella. Según el enfoque escogido se podrá explorar el HTML con *parseadores* en mera búsqueda de nuevos enlaces o se tendrá que extraer información del contexto para la indexación posterior y para puntuar las nuevas URLs encontradas. Más adelante comentaremos uno de los algoritmos de puntuación de enlaces más relevantes a día de hoy, Page Rank creado por Larry Page y et al. y usado por Google Search.

En este proceso habrá que tener en cuenta varias cosas:

- Antes de añadir las URLs encontradas a la frontera es necesario someterlas a un proceso de normalización y canonización. Esto será importante para no descargar páginas que puedan estasr alojadas en dominios diferentes. Páginas webs grandes pueden usar numerosas direcciones IP diferentes con el fin de balancear cargas y el contenido podrá ser mirror de la principal. Las URLs extraídas podrán ser absolutas o relativas. El esquema de una URL absoluta será esquema://servidor/ruta de acceso/recurso, y en caso de encontrar URLs que no sigan este formato o el escogido como principal deberá ser normalizado para serlo con respecto a la dirección absoluta. Según la implementación podrá ser relevante añadir el numero de puerto para todas las canonizaciones o excluirlas para todas. La normalización de URLs incluirá la limpieza de las mismas. Una URL "sucia" puede resultar trampa para el crawler ya que podría crear un bucle de peticiones en el que quedarse atrapado. Una URL no normalizada podría ser: http://www.dirtywebpage.com/index.php?page=results y la misma URL limpia será: http://www. dirtywebpage.com/results/

- El stoplisting es una técnica que ya se ha mencionado previamente que es importante de cara a evitar que nuestro índice se llene de palabras que no añadirán información relevante. Esta implementación generalmente consiste en tener una lista de stopwords, que estará compuesta por palabras como "y", "es", "si", "el", "de", etc.

- Otro problema frecuente al parsear una web descargada es que no siempre el archivo HTML llegará limpio. Es muy probable que nos encontremos con un documento sin las etiquetas de más alto nivel como podrían ser ¡head¿ o ¡body¿ y también sin las etiquetas de cierre. Si se pretende parsear las webs descargadas siguiendo la estructura HTML con el fin de encontrar enlaces u otra información relevante en ciertas etiquetas de la página podrá ser necesario un proceso de reconstrucción previo, para el cual se usarán herramientas de "tidying up".

- Las conocidas como *spider traps* tendrán que ser también tenidas en cuenta. Podrá darse el caso en que de manera provocada o accidantalmente el crawler se meta en un ciclo de rastreo infinito provocado por varias páginas webs que se apuntan la unas a la otras y sin más enlaces salientes del ciclo. Otra trampa comúnmente encontrada será la de las URLs infinitas. Podremos encontrarnos con scripts que generen dinámicamente URLs a los que se les vaya concatenando al final de la URL ciertas palabras, sin que lleven a ningún nuevo sitio. Hay ciertos protocolos que se pueden usar como controlar la longitud de los enlaces y establecer un límite a partir del cual no recoger más URLs.

Paralelización del proceso

Para indexadores de gran escala, una implementación unihilo no tendrá sentido ya que rastrear miles o millones de páginas webs necesitaría una grandísima mejora de rendimiento para ser eficiente. A continuación vamos a explicar dos posibles implementaciones de crawlers con procesamiento paralelo.

En primer lugar, podemos encontrarnos con la estructura multihilo. Principalmente, en esta aproximación empezaremos con un numero especifico de hilos ya alocados. Cuando de haya resuelto el dominio buscado, cada hilo abrirá un socket cliente y se conectará al servidor donde esté alojado la página web. Se enviará la petición HTTP y se irá leyendo del socket hasta que se acabe la información enviada, finalmente cerrando el socket. La idea principal es que se llevará a cabo una implementación bloqueante de los sockets mientras estos estén en uso. De igual manera, los hilos compartirán la frontera y el historial y manejarán individualmente sus estados y pilas. El acceso a la información guardada puede resultar lenta si se lleva a cabo con accesos bloqueantes de lectura en disco. Habrá implementaciones que prefieran utilizar procesos individuales con el fin de evitar la posible corrupción de memoria en el caso en que uno de los hilos muriera.

Otra alternativa será usar sockests no bloqueantes con un controlador de eventos [18]. En este caso, las llamadas de *connect, send o recv* vuelven nada más ser ejecutadas y en su lugar se usará la llamada *select*, que monitorizará los diferentes sockets a la vez y les asignará un proceso cuando reciba alguna llamada de lectura o escrituran través de ellos. Cada socket dispondrá de una estructura de datos que mantendrá el estado del mismo, así como del hilo que este esperando la terminación de algún proceso. Cuando una llamada de *select* sea recibida esta incluirá un identificador del socket correspondiente y se usará el registro del estado para continuar el procesamiento.

2.3.3. Algoritmo de puntuación de URLs

La importancia de una página web es en gran parte un asunto que se define de manera subjetivo. No obstante, hay diversas maneras según las cuales se puede intentar medir de una manera objetiva la relevancia de una web. Uno de las primeras aproximaciones a este problema comparaba la web con el mundo de citaciones académicas. Una web podía ser asemejada a un trabajo científico, el número de trabajos que lo citan serían los enlaces entrantes que lo referencian *(backlinks, inedges)* y los trabajos que el propio documento cita serían sus enlaces salientes *(forward links, outedges)*. No obstante, hay varios motivos por los que esta técnica de medición no es adecuada. Primero de todo, el mundo académico se rige por unas fuertes medidas de calidad, mientras que en la web cualquier cosa publicada no tiene implícita un nivel mínimo de calidad o relevancia. En segundo lugar, un programa web podría generar numerosas páginas con enlaces que direccionasen hacia una web específica, lo que constituiría una manera fácil de engañar al sistema. Por último, puede darse el caso en que una página web no haya conseguido suficientes enlaces que la referencien y no por ello debería ser considerada menos relevante.

Page Rank

Page Rank se basa en la siguiente idea: una página tendrá una alta posición si la suma del rank de sus enlaces entrantes es alta. Con esta concepción podemos eliminar uno de los problemas anteriormente mencionados: con este algoritmo se cubren tanto las páginas que tienen muchos enlaces entrantes como las que tienen pocos o solo uno.

Sea u una página web. F_u será el conjunto de páginas a las que apunta u y B_u el conjunto de páginas que apuntan a u. Definiendo $N_u = |F_u|$, N_u será el numero de enlaces a los que apunta u. c será una constante de normalización.[19] La siguiente ecuación define la versión simplificada del algoritmo:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N(v)} \tag{2.1}$$

PageRank determina el Rank R de cada página individualmente. El Rank de la página u esta definido recursivamente por el rank de las páginas que llevan hasta u. El PageRank de las páginas F_i que apuntan a u no tiene una importancia uniforme en el PageRank de u, si no que el peso dependerá de numero de páginas N_i a las que apunte F_i. Esto implica que a cuantas más páginas apunte F_i menos se beneficiará u de que F_i le apunte.

Toscamente el algoritmo creado por Larry Page y Sergey Brin se define así. Una variante de este algoritmo se propuso posteriormente para rastrear 60 millones de páginas webs. Al poco tiempo crearon un sistema de búsqueda llamado Backrub que fue publicado online en 1997 como Google.

A día de hoy hay numerosas mejoras en algoritmos de este calibre, siendo este también un tema de mucho interés en el ámbito de *Search Engine Optimisation* (SEO). No obstante vemos un problema fundamental en este tipo de algoritmos, no son aplicables para la *Deep Web*. Como mencionábamos previamente, la estructura hiperenlaceeada de la web hace posible que se pueda intentar encontrar una manera de medir la relevancia de una página por los enlaces que la apuntan o con los que apunta. Este será un problema relevante para los indexadores de *Deep Web*, que deberán encontrar otros mecanismos con los que puntuar las páginas webs que indexen.

3

Indexación de la Deep Web

La *Deep Web* no proporciona un acceso a contenido igual de cercano y asequible que el que nos ofrece la web de superficie. No solo se encuentra el contenido oculto detrás de interfaces de búsqueda, si no que muy habitualmente se genera de manera dinámica y sobre URLs que serán reutilizadas para futuras consultas, lo cual impide su indexación. El desafío principal cuando se habla de indexar o rastrear ("to crawl") la *Deep Web* es la interacción con las interfaces de búsqueda, que generalmente en literatura se suele limitar a formularios web, aunque es perfectamente posible encontrarse con servicios webs o APIs. Para poder automatizar una herramienta que vaya a interactuar con formularios web se presentan varios problemas prioritarios. En primer lugar, se tendrá que identificar los diferentes componentes del formulario y entender el comportamiento de los *inputs* que lo componen. El segundo gran reto consistirá en generar modelos de consultas (*query templates*), que serán el conjunto de *queries* que se decida para enviar a los formularios. Los modelos deberán intentar ser lo más eficientes en coste y eficacia, es decir, conseguir la máxima cobertura con el mínimo coste. A continuación veremos varios enfoques que abordarán los problemas presentados.

3.1. Alcance de la indexación

Nos encontramos en la literatura diversos modelos de indexación que se diferenciarán según el objetivo de alcance que persigan. Según esta taxonomía vamos a poder encontrar dos grandes conjuntos. En primer lugar están aquellos que persiguen una maximización de contenido en anchura, intentando abarcar el máximo numero de páginas diferente pertenecientes a la *Deep Web*. En segundo lugar están los indexadores tópicos o enfocados. Es estos casos se prioriza poder conseguir el máximo contenido de una misma página web, o de varias cuyo contenido y estructura sea similar. Veremos a continuación, que las técnicas de rastreo van a ser bastante diferentes para los dos casos.

En *Google's Deep Web Crawl* [20], se explica por qué la prioridad es conseguir indexar el numero mayor posible de "deep" webs posibles y no tanto conocer unas pocas pero con mucha profundidad. Google tienen *crawler*s que usa constantemente para webs de superficie, a los cuales si les das como semilla un par de páginas traídas desde la *Deep Web*, estos podrán intentar buscar información relevante dentro de estas buscando links etiquetados como "artículos similares" o "mostrar más", desde los cuales van a ir pudiendo llegar a nuevo contenido . También

argumentan que a nivel de costes no es coherente querer aspirar a extraer mucho contenido de un solo dominio o temática, ya que se necesitará una customización grande del programa y generalmente es muy costoso poder reutilizar dicha estructura a otras páginas webs aunque sean de contextos similares, ya que las estructuras de los formularios y el contenido de las webs finalmente se construye de manera muy especifica.

El objetivo que se persigue es crear una estructura que pueda generar consultas para millones de formularios diferentes entre sí. De esta manera se podrá conseguir una buena cobertura de numerosas páginas de la deep web, aunque esta pudiese resultar algo incompleta; y conseguirla con pocos envíos por cada dominio pero consiguiendo que los resultados obtenidos sean relevantes y útiles para ser indexados.

Para *crawler*s que vayan a ser específicos de dominio, o de temática, se cambiará de estrategia y se intentará un tipo de búsqueda en profundidad. En cada iteración del *crawl* se analizará la información contextual de la página. Se evaluará si la página en cuestión es relevante a la temática buscada, y si no lo fuese se subiría una altura en el arbol de profundidad y se seguiría el rastreo bajando por otra rama. Este tipo de rastreadores necesitarán de herramientas adicionales para la evaluación semántica de la información. En [18] se precisaba de un usuario para que fuese corrigiendo, revisando y creando nuevas taxonomías. En [21] incluyen un clasificador Bayesiano, que estimará la distancia en número de links entre la página actual y posibles páginas relevantes. Y en [22] se usan técnicas de *machine learning* con el mismo fin. Es obvio que estas técnicas van a resultar en altos costes en entrenamientos para los clasificadores, y para un usuario seguir el ritmo del *crawler* será inviable. No obstante, según el objetivo del rastreo, técnicas especificas pueden ser muy útiles. Por ejemplo, si imaginamos que estamos realizando una búsqueda sobre "análisis numérico" podría ser que primero empezásemos en páginas de departamentos de matemáticas o informática y de ahí pudiésemos navegar hasta páginas de facultades en las que poder encontrar trabajos o apuntes que contuviesen las palabras buscadas. En un buscador ajeno al contexto, la página inicial del departamento de matemáticas será puntuada a la baja, mientras que un buscador contextual podrá entender que la distancia en clicks a páginas relevantes es muy bajo y por consiguiente priorizará el rastreo de este dominio.

3.2. Frameworks para la selección de *queries*

La interacción con las interfaces de búsqueda de las páginas que se vayan a rastrear en la *Deep Web* se reducirá a enviar valores a los formularios web. El principal reto que nos encontramos es cómo generar un conjunto de consultas que devuelvan respuestas lo más eficientes en cuanto a coste y que proporcione el máximo contenido. A continuación vamos a ver varias técnicas repetidas a través de la literatura que exponen diferentes estructuras de procesos de generación de *query templates*. Los *query templates*, o modelos de consultas, serán el conjunto final de consultas hechas al formulario y se buscará que este conjunto genere contenido de respuesta lo más amplio posible. Los tres frameworks que se definen a continuación son en primer lugar, el que se basará en una muestra inicial para construir el modelo de consultas, *"Sampling based"*, en segundo lugar, el que seguirá un proceso incremental para construir su *query templates*, *Ïncremental based"* y en último lugar el que se basará en conocimiento previo, *.ºntology based"*.

3.2.1. Basado en muestreo- *Sampling based*

Una de las aproximaciones que comentábamos previamente, y de las más usadas en literatura, se basa en extraer del dominio web que se pretende rastrear una muestra o *sample* de páginas que pertenezcan a la misma web[12]. De este conjunto se recogerá una lista de *keywords*, o palabras clave, términos, y se calculará su frecuencia de aparición en la muestra o se le asignará

una puntuación basada en parámetros específicos. Los frameworks explicados a continuación se han ordenado en función del coste.

Naive one-step sampling

En primera instancia encontramos con la estructura de funcionamiento más *inocente* y menos costosa. Este algoritmo se compone de dos fases: habiendo encontrado la interfaz de búsqueda, se usan unas *queries* de prueba, "*probes*", para conseguir una muestra de resultados. De ahí se construye una lista de *keywords* según la frecuencia de aparición en los documentos extraídos. La segunda fase consistirá en iterativamente ir generando *queries* usando las palabras con frecuencias más alta y se irá determinando el incremento de cobertura que conseguirá cada query. Cuando se haya desarrollado el conjunto de *queries* final, se enviarán a la interfaz y se recuperarán los resultados de la bse de datos. En [23], se prueba este algoritmo y se llega a resultados del 90 % de cobertura en ciertos dominios considerados. Este mismo trabajo propone varias condiciones con las que se pararía la iteración. Se establece un limite de máximas *keywords* que se almacenarán y un máximo de envíos que se ejecutarán. También se trata con el concepto de "stop wrords", es decir, el filtrado de ciertas palabras que no añaden información antes de procesar el lenguaje natural.

Algoritmo avaricioso

Otro enfoque para rastrear páginas de *Deep Web* es en vez de simplemente buscar la máxima cobertura, perseguir la mínima tasa de superposición. Este abordaje consistirá en generar un conjunto de *queries* que al ser respondidas contengan la mínima información duplicada, por consiguiente la máxima cobertura con el mínimo numero que consultas realizadas.

La estructura de funcionamiento de estos *crawler*s seguirá el siguiente patrón: primero se descargará una muestra de la página web, de la que se obtendrán *keywords*. Con estos *keywords* se formará una *"query pool"* y de ella se escogerán un conjunto de consultas que al ser enviadas cubran toda la muestra descargada. Con el envío de dichas consultas a la base de datos original se obtendrá una cobertura buena de la base de datos entera, habiéndose enviado menos consultas que con otros algoritmos.En [24] defienden que:

1. Las *keywords* aprendidas de la muestra cubrirán gran parte de la base de datos original.

2. El grado de superposición en la muestra podrá extrapolarse a la base de datos original

3. Que la muestra y la query pool no tienen que ser grandes de tamaño para poder cubrir gran parte de la base de datos original.

No obstante, los mismos autores en [25] establecen que hay que tener en cuenta que al rastrear la *Deep Web*, los documentos obtenidos de las consultas a las interfaces web no van a ser del mismo tamaño, por lo que el número de palabras no será el mismo en cada resultado, es decir, la reivindicación de que se podrá extrapolar los resultados encontrados para la muestra a la base de datos original pierde fuerza, ya que la distribución de contenido en la *Deep Web* no seguirá una distribución uniforme. Proponen un algoritmo mejorado teniendo en cuenta el peso de los documentos a recoger y en el que se prioriza poder enviar la query con mayor grado de requerimiento lo antes posible. Similarmente, encontramos un algoritmo muy usado para medir la importancia de las palabras en una página en comparación con su importancia global, TF-IDF[26]. TF (*"term frequency"*) medirá la frecuencia de las palabras en la página de la que se ha extraído. IDF (*ïnverse document frequency"*) mide la importancia de la palabra entre todas las posibles páginas. Siguiendo su funcionamiento, eliminaríamos de la lista de posibles *keywords* para generar *queries* las que acarreen un valor alto de TF, ya que indicará que usar esta palabra para una query posiblemente resulte en un mayor numero de resultados duplicados.

3.2.2. Basado en incremento- *Incremental based*

Otra aproximación al rastreo de *Deep Web*s es la estructura incremental. En este tipo de *crawler*s, el ciclo de funcionamiento se convierte en un proceso iterativo. En primer lugar, se consigue una muestra de páginas de la web en cuestión. Se usará una estrategia especifica para seleccionar la siguiente mejor query y se enviará. El resultado en este caso será descargado y unido a la muestra previa, y con los nuevos datos adquiridos se volverá a analizar la muestra y a generar la mejor query ahora incluyendo los documentos recién descargados. Este tipo de frameworks son costosos y en la práctica no serán muy aplicados ya que habrá que añadir al coste ya existente de envío de consultas, el coste de descarga de documentos y de integración con los de la muestra, así como el de analizar reitarademente la muestra para conseguir nuevas *queries* pero teniendo en cuenta no repetir las ya enviadas, etc.

3.2.3. Basado en ontología- *Ontology based*

Uno de las soluciones más directas para rastreadores enfocados a dominios se basa en crear una ontología, o conocimiento previo sobre el contexto. Encontraremos que muchos trabajos van a conseguir de esta manera unos resultados óptimos de rendimiento al no tener que realizar el proceso de incluir la información descargadas de las muestras o iteradamente si se decidiese por una estructura incremental. Este tipo de frameworks necesitarán, no obstante, de un trabajo funcional y manual previo costoso. Será habitual también encontrar híbridos entre algunas de las taxonomías anteriormente mencionadas.

3.3. Procesamiento de formularios HTML

Habiendo decidido un framework para la selección de *queries*, nos encontramos con el problema de interactuar con los formularios webs de las interfaces de búsqueda. En *Google's Deep Web Crawl* [20], explican como se trae a flote o a la superficie (*"to surface"*) contenido de la *Deep Web* eligiendo adecuadamente un conjunto de valores para rellenar los formularios.

3.3.1. Peticiones HTTP

Cuando se rellena un formulario y se envía, el navegador envía un *HTTP request* con los *inputs* y sus valores rellenados. Se pueden hacer dos tipos de peticiones: GET y POST. Con GET los valores usados para rellenar el formulario son concatenados a la URL que envía el HTTP request, creando urls únicas. Al hacer POST los valores de entrada se envían en el cuerpo de la petición HTTP y la URL solicitada suele ser genérica y usada para múltiples peticiones, por lo que no genera contenido adecuado para ser indexado.

3.3.2. Tipos de entradas

Los formularios webs están compuestos por entradas en el archivo, llamadas *inputs*. Cada entrada estará definida por un *input tag* que a suvez tendrá un nombre (no necesariamente el que vaya a ver el usuario) pero que puede dar información contextual si se recoge. Existen también los *hidden inputs* que suelen ser para dar más información al servidor sobre la procedencia de la petición, sesión de usuario, etc. En [20] se prioriza las entradas de texto libre o en menus de selección y serán estas las que se rellenen para enviar consultas.

Se pueden categorizar los inputs según su funcionalidad. En primer lugar tendríamos entradas selectivas *(selection inputs)* y en segundo lugar entradas de presentación *(presentation*

inputs). Las primeras imponen condiciones de selección para el contenido. Pueden ser o un input de texto libre o un menú con opciones entre las cuales elegir. Habitualmente se intentará usar un *"wild card value"* o comodín: en los de texto libre se probará a insertar "", la cadena vacía, y en los menús se intentará buscar, si lo hay, un valor por defecto. Los inputs de presentación son los que suelen tratar con la ordenación o visualización del contenido. Cuando se aplique el algoritmo de crear *query templates* se va a intentar que los formularios que se vayan a enviar no contengan inputs de presentación obligatorios, ya que los sucesivos resultados que se generen tendrán contenido muy similar aunque presentado de manera diferente.

3.3.3. Evaluación de *query templates*

Un *query templates* (modelo/modelaje de consulta) será el conjunto de posibles consultas que se generan para enviar a un mismo formulario. Para evaluar si el modelo es bueno, en *Google's Deep Web Crawl* se aplica el **test de diferencia** *(informativeness test)*. Con el fin de poder mantener los objetivos expuestos de alcanzar el máximo numero posible de páginas de la *Deep Web*, no se utilizarán paseadores de contenido específicos al dominio, por lo que la diferenciación entre páginas será puramente léxica (en vez de semántica). Se observarán los resultados obtenidos por las consultas enviadas y si las páginas entre si se presentan sin suficientes diferencias se podrá asumir que es por alguna de las siguientes causas:

1. El modelo presentado está usando un input de presentación y por eso múltiples páginas generadas tienen un contenido similar. 2. Se están generando consultas rellenando demasiados inputs, por lo que al ser *queries* tan específicas no están produciendo resultados y por consiguiente las páginas generadas son muy similares entre sí. 3. Se esta generando algún tipo de archivo de error.

Entonces un modelo será "informativo" cuando los documentos devueltos en el envío de las posibles consultas sean suficientemente diferentes entre si. Para evaluar las páginas se usa un algoritmo que pueda:

1. Ser indiferente al formateo de los HTML (ya que los inputs de presentación generalmente lo que suelen es cambiar la distribución del HTML). 2. Ser tolerante a diferencias menores en las webs (por ejemplo los anuncios que ocupan un lugar en el HTML son partes de la web que irán cambiando). 3. No tener en cuenta los lugares donde aparecerían los valores insertados en los *inputs* a la hora de comparar páginas. Por ejemplo, una página de error podrá mostrar "No hay resultados para coches ¡color¿ del año ¡año¿", por lo que dos *queries* con los valores rojo"2016z .ªzul"2018", podrán ser consideradas diferentes cuando en realidad son una misma página de error. 4. Ser capaz de identificar crecimientos monótonos. Por ejemplo, habrá *inputs* que indiquen cuántos resultados mostrar por página. Al ir iterando sobre los posibles valores se verá que el contenido es idéntico menos por nuevo contenido concatenado al final.

3.3.4. Generación de valores de entrada para los *inputs*

Dentro de los *inputs* de texto libre encontraremos dos categorías. La primera será en la cual las frases o palabras insertadas sirvan como filtro textual de resultados que se extraerán de la base de datos. En segundo lugar tendremos *inputs* que precisen de valores determinados porque delimitarán el ámbito de selección. Por ejemplo, precio o código postal serán condiciones que limiten el ámbito de búsqueda en la base de datos. Para poder interactuar con ambos tipos de *inputs* se necesitarán definir estrategias diferentes, así como una para aprender a distinguirlos.

En [20] exponen que para los *inputs* genéricos, en primer lugar pensaron en la posibilidad de hacer una fuente de palabras para varios diferentes contextos. Rápidamente se puede descartar esa opción, ya que hay miles de contextos diferentes y para un buscador de la magnitud de Google

no solo tendría que generar esos conjuntos de palabras para miles de dominios, si no que también tendría que hacerlo en numerosos idiomas también. Claramente, esta aproximación, que podría ser válida para trabajos con objetivos más delimitados, no lo es para ellos. Pos consiguiente, crean inicialmente una fuente "semilla" de palabras genéricas para todos los dominios como valores para el input. Se ejecutan consultas con dichas palabras y con las respuestas recibidas se va actualizando el conjunto de valores.

3.4. Literatura relacionada

A continuación se va a describir el funcionamiento particular de varios trabajos sobre indexación de contenido de la *Deep Web* y sus soluciones a los diferentes problemas que se han venido presentando ya que añaden una visión práctica sobre diferentes enfoques.

DIADEM

En *DIADEM: thousands of websites to a single database*[3], Furche et al. desarrollan un Sistema de extracción automática capaz de extraer datos estructurados de diferentes dominios con alta precisión. En la herramienta combinan la exploración de páginas webs, identificación de información relevante y la inducción de wrappers exhaustivos. El principal obstáculo superado es poder llevar a cabo todo esto de manera automatica y presentan su solución, basada en el uso de la combinación de conocimientos fenomenológico y ontológico.

Su propuesta caerá dentro de dos taxonomías que hemos definido previamente. En primer lugar, pese a exponer que su sistema es eficaz en numerosos dominios, definen su alcance como enfocado[4], es decir, no conseguirá extraer información de páginas webs en contextos que no pertenezcan a su ontología. En segundo lugar, en la modalidad de *crawler* se corresponde con la que hemos definido como*incremental based*. DIADEM partirá de una base de conocimientos a través de los cuales comenzará el proceso de extracción, pero conforme vaya extrayendo datos de las webs que rastree, irá incluyendo dicha información a su ontología.

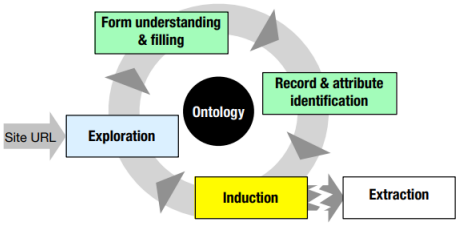

Figura 9: Arquitectura DIADEM[3]

De este trabajo es especialmente interesante justamente eso, el uso de la ontología. En la figura **??** se esquematiza la arquitectura de DIADEM. Comenzando con una única URL, DIADEM usa una combinación de *crawler*s enfocados, o específicos de dominio, junto con rellenado automático de formularios con el fin de provocar cambios y generar información. Con los eventos que se vayan encontrando se inducirán *wrappers* que serán los encargados de identificar y extraer el contenido relevante de la página web. La arquitectura está planteada de manera que cada componente se convertirá en un *relational transducer*, o lo que es lo mismo, en una caja negra que usará la secuencia de relaciones de entradas provenientes del componente anterior para generar una serie de relaciones salientes que enviar al siguiente.

DIADEM tiene un componente que se encargar de descubrir los *query templates*s de manera específica pro dominio. La manera en la que lo consigue es primero de todo encontrando

patrones en páginas webs, independientemente de su contexto. Encontrado el *template* en la página web, entrará en uso la ontología según dominio. Uno de los puntos claves será encontrar atributos *pivote*, que serán atributos típicos, obligatorios y fáciles de identificar: por ejemplo en dominios de compra-venta de casa el campo "Precio". Este tipo de atributos también ayudarán a identificar patrones que puedan parecer regulares pero que aporten información irrelevante, como anuncios.

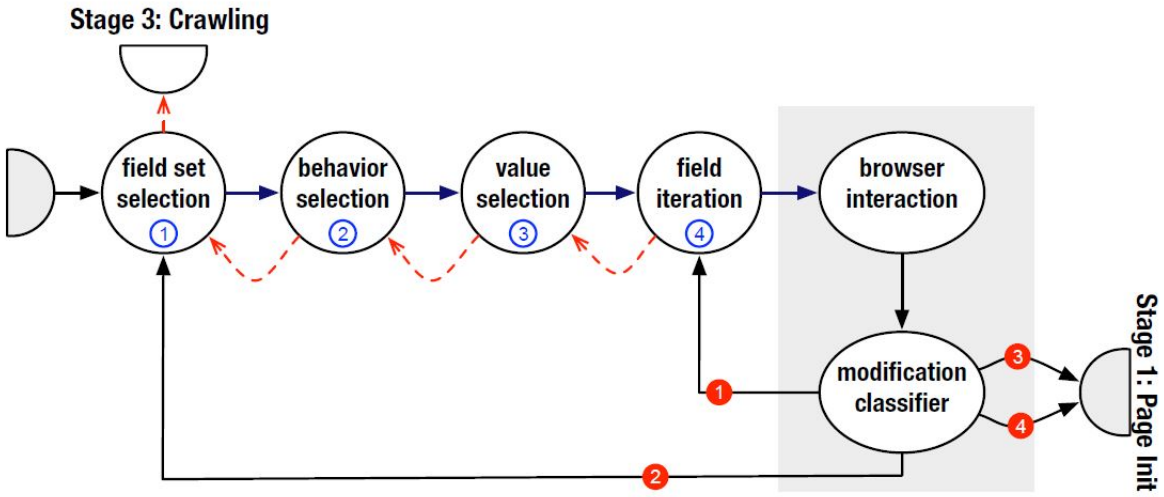

Figura 10: Sub-red de rellenado de formularios de DIADEM[3]

En contra de como se ha visto que Google plantea el rellenado automático de formularios, Furche et al. usan una red de transductores conscientes de dominio que usan su conocimiento para rellenar las entradas de los formularios y para reaccionar ante las respuestas del envío. En la figura XX vemos un esquema de la red. Hay cuatro componentes principales:

1. Selector de campos a rellenar.
2. El selector de comportamiento de los campos elegidos, por ejemplo, un campo de texto será tratado como un *autocomplete*.
3. El rellenador de valores para los campos.
4. El iterador sobre los diferentes campos.

Además, hay dos transductores adicionales que se encargarán uno de interactuar con el navegador, enviando las consultas, y el otro de identificar diferentes cambios en la página web tras ejecutar la consulta. Se ejecutarán los siguientes cambios de estado basándose en:

1. Si no se han registrado cambios sustanciales se continua con la iteración sobre los campos.
2. Si se registran cambios en los estados de las entradas del formulario se pasará a re-elegir los campos a rellenar.
3. Si se llega a una nueva página se acaba la fase de rellenado.

Furche et al. introducen junto con DIADEM un lenguaje extensión de XPath para interactuar con aplicaciones webs y extraer contenido de ellas, OXPath[4] . OXPath presume de poder

escalar según se vayan rastreando nuevas páginas webs mientras que permanece en complejidad polinomial. Aunque se limitan a la extracción de datos de la *Deep Web*, esta herramienta permite interactuar con páginas webs de complejidad diversa simulando las acciones de un usuario. Un ejemplo de su uso se puede ver en la figura 11.


```
doc("amazon.co.uk")
  //field()[@title='Search in']/{"Books"}❶
  /following::field()[@title='Search for']/{"Seattle" /}❷
  //field()[@alt='Go']/{click /}❸
  //a[*.refinementLink[.~'History']]/{click /}❹
  //*.result:<book>[.//a.title:<title=(.)>/{click /}❺
  //b[.~'Publisher']/following-sibling::*:<publisher=(.)>]
  [.//span.price[1]:<price=(.)>]
```

Figura 11: Ruta OXPath en Amazon extraído de [4]

ANDES

En [5] nos encontramos con una metodología diferente propuesta para extraer datos webs basado en el lenguaje XML, que compone el *backbone* (columna vertebral) de varios sistemas de extracción de datos en uso en producción en IBM. Un esquema de la arquitectura simplificada de ANDES lo podemos ver en la figura 12. Principalmente, el esquema de funcionamiento seguirá los siguientes pasos:

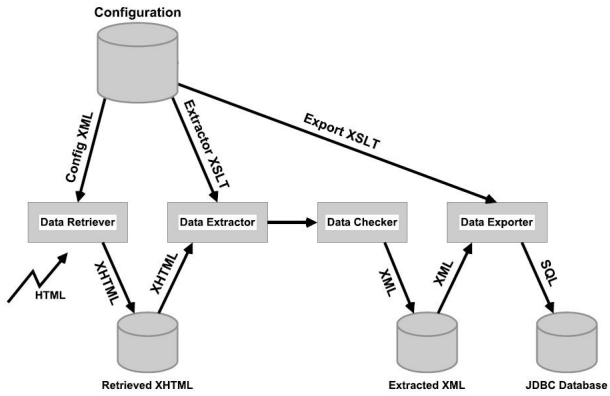

Figura 12: Arquitectura de ANDES extraído de [5]

1. El HTML de las páginas webs es descargado.
2. Las webs son trasladadas al extractor que llevará a cabo la extracción de los datos y síntesis de la estructura.
3. Los documentos XMLs generados por el extractor son enviados al *data checker* y finalmente al *exportador*.
4. El exportador inserta los datos XMLs a una base de datos relacional.

En ANDES proponen que el primer paso para la extracción de datos de webs sea la traducción del archivo HTML a un documento XML bien formado sintácticamente. Primeramente, se pasará el HTML por un filtro XML, generando lo que hoy se conoce como un XHTML. Este será filtrado a su vez por varias extensiones de XSLT, consiguiendo finalmente un XML limpio y robusto para la extracción de contenido estructurado donde antes no lo había.

Uno de los principales problemas que se encuentra en la extracción de datos de HTMLs en trabajos relacionados es que los mecanismos comienzan a fallar en el momento en que la estructura de la página web cambia, debido a que los *wrappers* usados tienden a usar rutas absolutas a través de XPath. ANDES propone la búsqueda de *anchors*, o anclas, en las páginas, que sean los puntos de inicio de la extracción de los datos y en la mayoría de veces estas anclas se encontrarán en el contenido de la página y no en la ruta HTML.Por ejemplo, en una web de dominio de compra-venta de casas el precio de una casa que se este visualizando podrá encontrarse en la segunda etiqueta de texto plano después de encontrar el título del producto, produciéndose una ruta XPATH similar a la siguiente: /HTML/BODY/TITLE/P[2]. Si la siguiente página que viésemos cambiase de estructura no seríamos capaces de encontrar el precio siguiendo la ruta establecida. No obstante, buscando el *anchor* probablemente todos los precios de compra-venta de las casas se encuentren en la proximidad de la palabra "Precio"que podremos encontrar buscando en el contenido.

Podemos ver que este sistema no sigue un framework incremental, pretende desde su conocimiento del dominio poder identificar estas anclas y responder con *keywords* específicos que ya pertenecen a su ontología. Como hemos ido viendo, los sistemas que usan ontologías se limitan a ser indexadores enfocados a dominios.

3.4.1. Rastreadores en la Dark Web

A continuación vamos a explicar cómo los siguientes trabajos han llevado a cabo un rastreo u indexación en la red Tor. Se definirán cuáles son los principales problemas que se encuentran en estos desarrollos y qué soluciones se proponen.

Crawler para extraer datos de Silk Road

En redes oscuras como Tor encontramos que han emergido numerosos mercados online que dificultan a las fuerzas del orden la identificación de los compradores y vendedores. Como ya hemos mencionado antes, este tipo de mercados tienden a especializarse en el mercado negro. El trabajo Traveling the Silk Road fue publicado meses antes del cierre a manos del FBI del conocídisimo Silk Road, o Camino de la Seda. Silk Road proveía de una infraestructura montada sobre un servicio oculto que aseguraba el anonimato de ambos vendedor y comprador. Para entrar al sitio debías crearte una cuenta y podías acceder a dos tipos de productos: los publicados públicamente o los que solo se vendían en listas cerradas a las que necesitabas permiso. Los pagos se realizaban a través de PayPal y con el método del *escrow*, o deposito en garantía. El vendedor no recibía su dinero, que lo mantenía un intermediario y cedía una comisión a Silk Road, hasta que el comprador no "finalizaba"la compra.

Los objetivos propuestos por Nicholas Christin eran realizar una medida comprensiva y analítica sobre Silk Road [6], sobre los operadores y usuarios y sobre los productos que se ofrecían Para realizarlo se desarrolló un rastreador que recogió datos durante 8 meses. El procedimiento seguido fue el siguiente:

- Usando una cuenta previamente registrada se accedía a la web. Descubrieron que se precisaba pasar un captcha pero que podía ser saltado ya que las cookies de sesión duraban una semana y si se actualizaban manualmente el *crawler* podía acceder a la web.
- Descubrieron que Silk Road publicaba sus artículos en URLs que incrementaban linearmente su valor entero. De esta manera consiguieron rastrear las páginas de usuarios, categorías y artículos.
- Se aseguraban de que las conexiones se hiciesen siempre o lo más a menudo posible a través de circuitos nuevos. Es importante que el objeto de un estudio no conozca que está siendo estudiado ya que podría cambiar su funcionamiento.

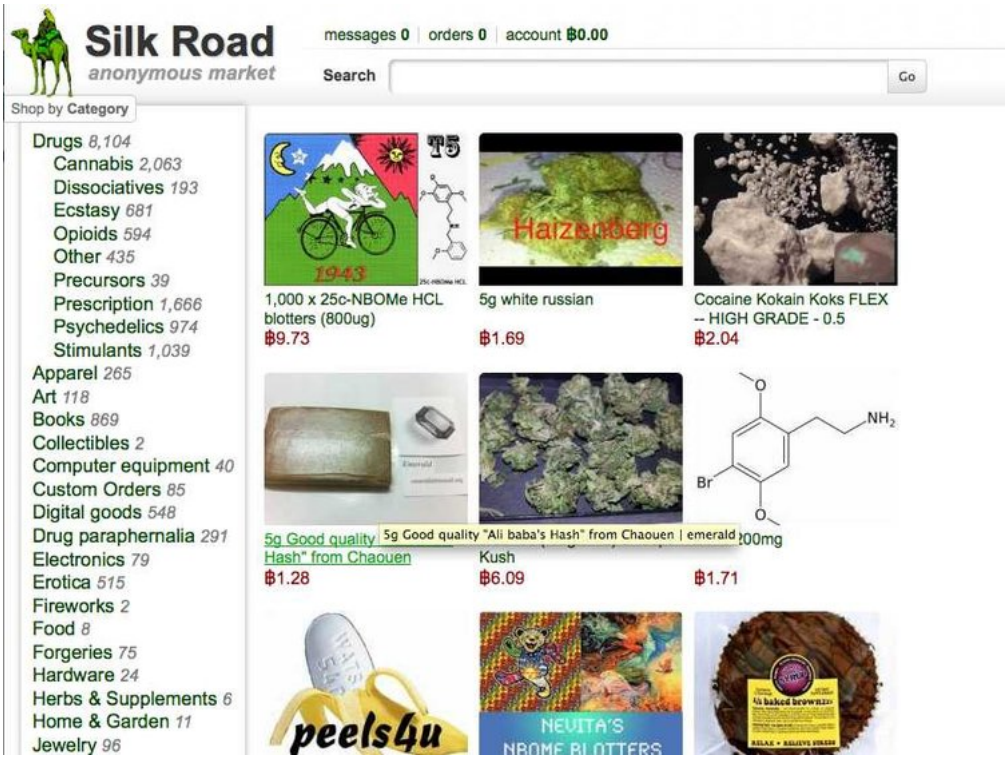

Figura 13: Página principal de Silk Road extraída de [6]

Este trabajo está claramente enfocado en un único dominio. Aunque no se específica con detalle cómo se realizó la extracción de datos se puede ver que hay una gran carga ontológica. El conocimiento del contexto es lo que hizo que el estudio consiguiese unos buenos resultados en cuanto a información recuperada y al post-analisis hecho.

Tor Browser crawler

En [27], Juarez et al. desarrollan un *crawler* para Tor con el fin de recuperar datos de red con los que demostrar que cierto ataques de *Website Fingerprinting* no resultaban tan eficaces como se asumía. Para ello desarrollan un *crawler* que usan junto con herramientas de captura de paquetes de red y empiezan a rastrear una lista de páginas que cogen de una lista preconfigurada. Este *crawler* tiene un objetivo claramente diferente a otros que hemos visto. En este caso, en vez

de querer extraer información de las páginas visitadas se busca poder recuperar información de red sobre usuarios conectados. El uso de herramientas externas, como *dumpcap*, que se encargan de este fin, también quita un modulo de funcionalidad relevante al rastreador, que en este caso se limita a ser el transporte.

BUbiNG crawler

En [28], usan el crawler desarrollado en [29] con el fin de llevar a cabo un estudio analítico y estadístico sobre Tor: identificar la topología de los *servicios ocultos* en Tor y la relación que pudiese tener con el contenido de las diferentes páginas.

El crawler estaba diseñado de manera que seguía una búsqueda en anchura, con una cola de prioridad *FIFO* como frontera y con dos modulos separados encargados de la exploración y de la recogida de datos. Para su uso se necesitaba una lista de URLs semilla, y los hilos iban extrayendo los enlaces y exportando el contenido de los datos descargados a memoria. Seguidamente un hilo encargado del *parseado* analizaba el contenido buscando posibles enlaces para visitar. Finalmente, se comprobaba que los enlaces encontrados no habían sido ya visitados y que eran válidos antes de añadirlos a la frontera.

Podemos ver que esta estructura sigue la de un rastreador común y no usa métodos de indexación en *Deep Web*. La motivación de ello es que uno de los objetivos principales del trabajo consiste en analizar el grafo de servicios ocultos de Tor. Específicamente se analizan tres tipo de relaciones, el numero de enlaces entre páginas, el numero de enlaces entre dominios y el numero de enlaces entre servicios. Uno de los principales limitaciones de este tipo de implementaciones será la dimensión de páginas que se incluyan en la semilla, ya que muchos servicios ocultos deliberadamente limitan el numero de enlaces entrantes con el fin de permanecer ocultos. Una solución propuesta es un programa que genera aleatoriamente direcciones .onion y prueba su existencia.

Otras herramientas

A continuación se listan una serie de herramientas que actualmente se encuentran indexando servicios ocultos de Tor. Por falta de documentación no tenemos manera de conocer su estructura o funcionamiento pero resultan igualmente trabajos de interés:

- Onion Scan: desarrollado por Sarah Jamie Lewis, Onion Scan es una herramienta que pretende ayudar a los operadores de *hidden services* encontrar vulnerabilidades en sus *sites*, así como ayudar a investigadores a poder llevar un registro de páginas de la *Dark Web*. Estos dos objetivos pretenden ayudar a concienciar sobre la necesidad del uso de estas tecnologías que preservan nuestra privacidad[1].

- TorBot: crawler en *Python* que rastrea páginas con una estructura de funcionamiento de rastreo en *Surface Web* pero desarrollada para funcionar en *onion sites* [2].

- Ahmia: fundado por Juha Nurmi y subvencionado en parte por Google Summer of Code, Ahmia aspira a ser un buscador puntero en la red Tor, acercando contenido a los usuarios menos expertos en este tipo de redes. Hace un prefiltrado de su información y elimina referencias a sitios que incluyan contenidos abusivos[3].

- DuckDuckGo: Por último, DuckDuckGo es un buscador de contenido que ha sido integrado en TorBrowser por sus reivindicaciones de no recoger ningún dato de usuario, manteniendo en todo momento su anonimato, ni de mostrar anuncios. Aunque no se corresponde a la

[1] https://onionscan.org/
[2] https://github.com/DedSecInside/TorBoT [21 de junio, 2018]
[3] https://ahmia.fi/about/ [21 de junio, 2018]

misma categoría que los anteriores trabajos mencionados es relevante mencionar que este navegador indexa webs tanto de la superficie como *onion sites*[4].

[4]https://duckduckgo.com/ [21 de junio, 2018]

4

Conclusiones y trabajo futuro

Este trabajo ha proporcionado una perspectiva amplia y completa sobre el área de minería de información, sobre la cual al comenzar el mismo partía con las mínimas nociones básicas. Se ha visto como se han desarrollado las tecnologías de extracción de información y por qué vienen motivados estos cambios. Principalmente, se han tratado varias clasificaciones de indexadores para la Deep Web, según alcance y según estructura de funcionamiento y hemos podido ver que aunque en primer lugar pudiesen parecer taxonomías independientes, finalmente en la práctica resultan muy estrechamente relacionadas.

En segundo lugar también hemos adquirido unas buenas nociones teóricas y prácticas sobre las *DarkNets*, en especial sobre los servicios ocultos de Tor. Hemos visto que la extracción de datos en páginas contenidas en la Deep Web cobra mayor complejidad en estos entornos. La mala fama que precede a estas redes ofuscan su origen humanitario y sus características de privacidad y anonimato que más poco a poco vamos exigiendo en el resto de los ámbitos de nuestras vidas. Hemos visto que estas tecnologías usadas bien son inofensivas y que nos proporcionan otro mundo nuevo dentro del cual hay cuantiosa y valiosa información por ser descubierta.

Como comentábamos al principio de este trabajo, la ciberinteligencia abarca una temática muy amplia que se centra en la creación de inteligencia a partir de información extraída de Internet, incluyendo las partes de la red a las cuales hasta hace poco los rastreadores no eran capaz de llegar. Ahora nos encontramos ante un mundo de posibilidades y una cantidad creciente de información disponible ante nuestras manos. El desarrollo de indexadores de contenidos para la Deep Web cambia considerablemente en cuanto a las metodologías usadas para indexar la *Surface Web*.El propósito de este trabajo ha cumplido con parte de este objetivo, aprendiendo cómo poder acceder a información oculta en la Deep Web y cómo se puede extraer. Y finalmente, hemos visto también que las aplicaciones prácticas de la indexación de contenido dista de tener un único objetivo de crear servidores de búsqueda.

Como trabajo futuro se propone el diseño e implementación propia de un indexador de contenido para la Deep Web haciendo uso de las metodologías y herramientas aprendidas durante el transcurso del desarrollo de este trabajo. Se propone crear una herramienta en lenguaje *Python* a través de la cual se pueda rastrear páginas pertenecientes a la *Deep Web* y servicios ocultos alojados en *onion sites*. Se propone desarrollarlo en *Python* por diversas razones encontradas a lo largo de este trabajo. En primer lugar, porque existen numerosas herramientas y librerías con las que poder interactuar con la red Tor, montar sencillamente un cliente HTTP

con el que poder enviar y solicitar peticiones a páginas webs y *parsear* sencillamente páginas webs descargadas. La interacción con los formularios webs se presenta como el problema que dará más complicaciones. Por último, se ha visto que numerosos trabajos de literatura reciente han desarrollado herramientas en *Python* por las facilidades que ofrece este lenguaje para su ejecución y portabilidad.

Según los objetivos que se propongan podremos definir el alcance del sistema y el *framework* de desarrollo. El interés personal de la autora de este trabajo suscita ganas de avanzar más en el conocimiento sobre la minería de datos a través de indexadores y especialmente en los ámbitos de las *Deep Webs* y *DarkNets*.

Glosario de definiciones

- **Deep Web**: parte de la web cuyas páginas y servicios no están enlazados a través de hiperenlaces, por lo que los indexadores de contenido no pueden acceder a ellos a través de los métodos tradicionales de rastreo web.
- **Surface Web**: parte de la web que conocemos y que está hiperenlazada.
- **Dark Net**: un conjunto de páginas y servicios que pertenecen a la Deep Web y que a su vez se montan sobre redes de anonimato.
- **Ciberinteligencia**: la creación de inteligencia a través de la extracción, evaluación e interpretación de información y datos recabados a través de la web.
- ***Hidden service***: servicio oculto montado sobre la red Tor.
- **Onion site**: página web perteneciente al sub-dominio .onion y que necesita de un navegador Tor para acceder a su contenido.
- **Indexador**: sistema que procesa generalmente lenguaje natural y lo guarda de manera que la búsqueda y consulta de su contenido sea fácil y eficiente.
- ***Crawler***: módulo dentro de un indexador o independiente que rastrea la web. Generalmente, se entenderá que su función es encontrar páginas webs con el fin de extraer contenido de ellas.
- ***Parsear***: recorrer un archivo, documento u entrada de datos, identificando partes.
- ***Fingerprinting***: un tipo de ataque que sucede al supervisar trafico para identificar patrones que sean únicos de un mismo usuario o página web.
- **Red de baja latencia**: red que está optimizada para procesar grandes volúmenes de datos con mínima latencia (retardo).
- **Transductor**: Elemento que responde ante unas señales entrantes, como cambios de estado. Usado citando a [3].

Bibliografía

[1] Hal Burch and Bill Cheswick. Mapping the internet. *Computer*, 32(4):97–98, 1999.

[2] Users- tormetrics, directly-connecting clients. `https://metrics.torproject.org/userstats-relay-country.html`. [21 de junio, 2018].

[3] Tim Furche, Georg Gottlob, Giovanni Grasso, Xiaonan Guo, Giorgio Orsi, Christian Schallhart, and Cheng Wang. Diadem. *Proceedings of the VLDB Endowment*, 7(14):1845–1856, 2014.

[4] Tim Furche, Georg Gottlob, Giovanni Grasso, Christian Schallhart, and Andrew Sellers. Oxpath: A language for scalable data extraction, automation, and crawling on the deep web. *The VLDB Journal*, 22(1):47–72, 2013.

[5] Jussi Myllymaki. Effective web data extraction with standard xml technologies. *Computer Networks*, 39(5):635 – 644, 2002.

[6] Nicolas Christin. Traveling the silk road: A measurement analysis of a large anonymous online marketplace. *WWW '13 Proceedings of the 22nd international conference on World Wide Web*, pages 213–224, 2013.

[7] Domo. Data Never Sleeps 5.0 — Domo.

[8] David Westerman, Patric R. Spence, and Brandon Van Der Heide. A social network as information: The effect of system generated reports of connectedness on credibility on twitter. *Computers in Human Behavior*, 28(1):199 – 206, 2012.

[9] Julia Lane, Victoria Stodden, Stefan Bender, and Helen Nissenbaum. *Privacy, big data, and the public good: Frameworks for engagement.* Cambridge University Press, 2014.

[10] Google. How Google Search Works — Crawling & Indexing.

[11] Michael K. Bergman. White Paper: The Deep Web: Surfacing Hidden Value. *Journal of Electronic Publishing*, 7(1), aug 2001.

[12] Xu Sun. *Practical Guides for Data Retrieval in Deep Web Crawling.* PhD thesis, University of Windsor (Canada), 2016.

[13] Francisco Andreu Sanz. *TECNOLOGÍAS PARA LA PROTECCIÓN CONTRA LA PRIVACIDAD Y EL ANONIMATO.* PhD thesis, Universidad Autonoma de Madrid, 2018.

[14] A. Biryukov, I. Pustogarov, and R. P. Weinmann. Trawling for tor hidden services: Detection, measurement, deanonymization. In *2013 IEEE Symposium on Security and Privacy*, pages 80–94, May 2013.

[15] Eric Jardine. The Dark Web Dilemma: Tor, Anonymity and Online Policing.

[16] Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[17] Gautam Pant, Padmini Srinivasan, and Filippo Menczer. Crawling the Web.

[18] Soumen Chakrabarti. *Mining the Web- Discovering knowledge from hypertext data.* 2002.

[19] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[20] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's deep-web crawl. *Proceedings of the VLDB Endowment*, 1(2):1241–1252, 2008.

[21] M Diligenti, F M Coetzee, S Lawrence, C L Giles, and M Gori. Focused Crawling Using Context Graphs.

[22] Lu Jiang, Zhaohui Wu, Qian Feng, Jun Liu, and Qinghua Zheng. Efficient deep web crawling using reinforcement learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 428–439. Springer, 2010.

[23] Luciano Barbosa and Juliana Freire. Siphoning Hidden-Web Data through Keyword-Based Interfaces.

[24] Jianguo Lu, Yan Wang, Jie Liang, Jessica Chen, and Jiming Liu. An Approach to Deep Web Crawling by Sampling. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 718–724. IEEE, dec 2008.

[25] Yan Wang, Jianguo Lu, and Jessica Chen. Crawling Deep Web Using a New Set Covering Algorithm. pages 326–337. Springer, Berlin, Heidelberg, 2009.

[26] G. G. (Gobinda G.) Chowdhury. *Introduction to modern information retrieval.* Facet, 2010.

[27] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. A Critical Evaluation of Website Fingerprinting Attacks.

[28] Massimo Bernaschi, Alessandro Celestini, Stefano Guarino, and Flavio Lombardi. Exploring and Analyzing the Tor Hidden Services Graph. 11(4), 2017.

[29] Paolo Boldi, Andrea Marino, Massimo Santini, and Sebastiano Vigna. Bubing: Massive crawling for the masses. *ACM Transactions on the Web (TWEB)*, 12(2):12, 2018.